So Far (Schematically) yet So Near (Semantically)

Amit Sheth[1] and Vipul Kashyap[2]

[1]Bellcore, 444 Hoes Lane, Piscataway, NJ 08854-4182

[2]Department of Computer Science, Rutgers University, New Brunswick, NJ 08903

Abstract
In a multidatabase system, schematic conflicts between two objects are usually of interest only when the objects have some semantic affinity. In this paper we try to reconcile the two perspectives. We first define the concept of semantic proximity and provide a *semantic taxonomy*. We then enumerate and classify the *schematic and data conflicts*. We discuss *possible semantic similarities* between two objects that have various types of schematic and data conflicts. Issues of uncertain information and inconsistent information are also addressed.

1 Introduction

Many organizations face the challenge of interoperating among multiple independently developed database systems to perform critical functions. With high interconnectivity and access to many information sources, the primary issue in the future will not be how to efficiently process the data that is known to be relevant, but which data is relevant.

Three of the best known approaches to deal with multiple databases are tightly-coupled federation, loosely-coupled federation, and interdependent data management [SL90][She91a]. A critical task in creating a tightly-coupled federation is that of schema integration (e.g., [DH84]). A critical task in accessing data in a loosely-coupled federation [LA86, HM85] is to define a view over multiple databases or to define a query using a multidatabase language. A critical task in interdependent data management is to define multidatabase interdependencies [RSK91]. An additional approach is based on the paradigm of Intelligent and Cooperative Information Systems [PLS92] which involves exchange of information and expertise. In performing any of these critical tasks, and hence in any approach to interoperability of database systems, the fundamental question is that of identifying objects in different databases that are semantically related, and then resolve the schematic differences among semantically related objects. In this paper, we are interested in the **dual perspective** that emphasizes both the semantic similarities and the schematic

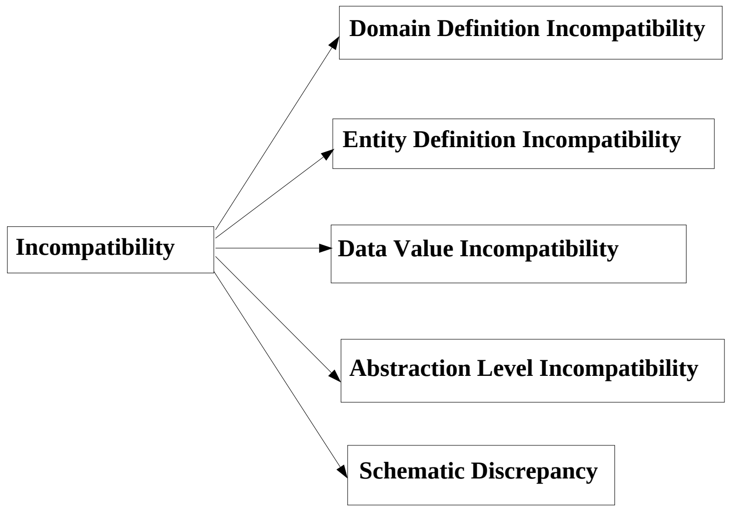
Incompatibility
Domain Definition Incompatibility
Entity Definition Incompatibility
Data Value Incompatibility
Abstraction Level Incompatibility
Schematic Discrepancy

Figure 1: Structural Incompatibilities due to Heterogeneity

(representational/ structural) differences.

While there is a significant amount of literature discussing schematic differences, work on semantic issues (e.g., [Ken91]) in the database context is scarce. Classification or taxonomies of *schematic differences* appear in [DH84, BOT86, CRE87, KLK91, KS91]. However, purely schematic considerations do not suffice to determine the similarity between objects [FKN91][SG89]. In this paper we try to reconcile the two perspectives. We develop a **semantic taxonomy** emphasizing semantic similarities between objects and show its relationship to a **structural taxonomy** emphasizing schematic (structural/representational) differences among the objects.

In section 2 we introduce the concept of semantic proximity that characterizes the degree of semantic similarity between a pair of objects. Understanding and representing semantic similarities and schematic differences between objects may involve understanding and modeling **uncertainty**, **inconsistency** and **incompleteness** of information pertaining to the objects (at both intensional and extensional levels), and the relationships between the objects. We address some of the issues of uncertainty and inconsistency. In section 3, we describe *fuzzy terminological relationships* [FKN91] by expressing the fuzzy strengths as a function of the semantic proximity between two objects. Section 6 addresses the *data value incompatibility problem* which arises out of the inconsistency between related data and the semantic similarities possible between inconsistent data.

The remaining sections deal with a broad class of schematic differences and the possible semantic similarities between the objects having those differences. Section 4 deals with the *domain incompatibility problem* [CRE87] which arises when attributes have different domain definitions. Section 5 discusses the *entity definition incompatibility problem* [CRE87] which arises when the entity descriptors used for the same entity are partially compatible. Section 7 deals with the *abstraction level incompatibility problem* [DH84] which arises when the same entity is represented at different levels of abstraction. Section 8 deals with the *schematic discrepancy problem* [KLK91] which arises when data in one database corresponds to schema elements in another.

2 Semantic Similarities between Objects

In this section, we introduce the concept of *semantic proximity* to characterize semantic similarities between objects, and use it to provide a classification of semantic similarities between objects.

We distinguish between the *real world*, and the *model world* which is a representation of the real world. The term object in this paper refers to an object in a model world (i.e., a representation or intensional definition in the model world, e.g., an object class definition in object-oriented models) as opposed to an entity or a concept in the real world. These objects may model information at any level of representation, viz. *attribute level* or *entity level.*[1]

Wood [Woo85] defines semantics to be "the scientific study of the relations between signs and symbols and what they denote or mean." It is not possible to completely define what an object denotes or means in the model world [SG89]. We consider these to be aspects of *real world semantics* (RWS) of an object[2].

Our emphasis is on identifying semantic similarity even when the objects have significant representational differences [She91b]. Semantic proximity is an attempt to characterize the degree of semantic similarity between two objects using the RWS. It provides a qualitative measure to distinguish between the terms introduced in [She91b], *viz. semantic equivalence*, *semantic relationship*, *semantic relevance* and *semantic resemblance.* Two objects can be *semantically similar* in one of the above four ways. Semantic equivalence is *semantically closer* than semantic relationship and so on.

2.1 A Model for Semantic Classification

Given two objects O_1 and O_2, the *semantic proximity* between them is defined by the 4-tuple given by

$$\mathbf{semPro(O_1, O_2) = <Context, Abstraction, (D_1, D_2), (S_1, S_2)>}$$

where D_i is domain of O_i and S_i is state of O_i.

A context of an object is the primary vehicle to capture the RWS of the object. Thus, the respective contexts of the objects, and to a lesser extent the abstraction used to map the domains of the objects, help to capture the semantic aspect of the relationship between the two objects.

2.1.1 Context of the two Objects

Each object has its own context. The term context in semPro refers to the context in which a particular semantic similarity holds. This context may be related to or different from the contexts in which the objects were defined. It is possible for two objects to be

[1]Objects at the entity level can be denoted by single-place predicates P(x) and attributes can be denoted by two-place predicates Q(x,y) [SG89].

[2]The term "real world semantics" distinguishes from the "(model) semantics" that can be captured using the abstractions in a semantic data model. Our definition is also intensional in nature, and differs from the extensional definition of Elmasri et al. [ELN86] who define RWS of an object to be the set of real world objects it represents

semantically closer in one context than in another context. Some of the alternatives for representing a context *in an interoperable database system* are as follows.

- In [SM91], the context is identified as the semantics associated with an application's view of existing data and is called the **application semantic view**. They propose a rule-based representation to associate metadata with a given attribute, and use this rule based representation to define the application's semantic view of the data.
- Just as a context may be associated with an application, it can also be associated with a **database** or a group of databases (e.g., the object is defined in the context of DB1).
- When many entities participate in a relationship, the entities can be thought of as belonging to the same context, which in this case is identified as the **relationship** in which the entities participate.
- In a federated database approach, we can use a **federated schema** [SL90] to identify a context to which two objects may belong to.
- From the five-level schema architecture for a federated database system [SL90], a context can be specified in terms of an **export schemas** (a context that is closer to a database) or an **external schema** (a context that is closer to an application). We can also build a context hierarchy, by considering the contexts associated with the external schemas to be subcontexts of the context associated with the appropriate federated schema.
- At a very elementary level, a context can be thought of as a **named collection** of the domains of the Objects.
- Sometimes a context can be "hard-coded" into the definition of an object. For example, when we have the two entities EMPLOYEE and TELECOMM-EMPLOYEE, the TELECOMMUNICATIONS context is "hard-coded" in the second entity. We are interested in representing and reasoning about context as an explicit concept.
- In our classification scheme, we are often interested in the cases where the context(s) of the objects under consideration can be determined to be one of the following. (In cases other than ALL and NONE, specific instances of semPro must name context(s) explicitly.)
  - ALL, i.e. the semPro of the objects is being defined *wrt* all possible contexts. The specific context need not be named.
  - SAME, i.e. the semPro of the objects is being defined *wrt* the same context. The context must be explicitly specified in an instance of a semPro.
  - SOME, i.e. the semPro of the objects are being defined *wrt* more than one context. The applicable contexts must be individually or collectively specified in an instance of a semPro.

  - SUB-CONTEXTS, when the semPro can be defined in a previously defined context that is further constrained. The subcontext must be specified in an instance of a semPro.
  - NONE, i.e. the objects under consideration do not exhibit any useful semantic similarity under any context.

Additional research is needed to identify appropriate representations of context, and develop a practical framework for semi-automatic ways of comparing and manipulating contexts (e.g., taking a union of two contexts). While it may not be possible to precisely define the context of an object, it may be useful to simply name it at a specific level of information modeling architecture (e.g., external schema or federated schema). A partial context specification can be used by humans to decide whether the context for modeling of two objects is the same or different, and whether the comparison of semantic similarity of objects is valid in all possible contexts or specific ones. Examples and discussions in the rest of the paper will clarify these points. An abstraction, discussed next, by itself cannot capture the semantic similarity, because it is always possible to construct a mapping between two semantically unrelated objects. However, if there is a semantic similarity between two objects, then we should be able to do so *wrt* a particular (or all) context(s).

2.1.2 Abstraction used to map the Objects

We use the term abstraction to refer to a mechanism used to map the domains of the objects to each other or to the domain of a common third object. Some of the more useful and well defined abstractions are:

- A **total 1-1 value mapping** between the domains of the objects, i.e., for every value in the domain of one object, there exists a value in the domain of the other object and vice versa. Also there is a one to one correspondence between the values of the two domains.
- A **partial many-one mapping** between the domains of the objects. In this case some values in either domain might remain unmapped, or a value in one domain might be associated with many values in another domain.
- The **generalization** abstraction to relate the domains of the concerned objects. One domain can generalize/specialize the other, or domains of both the objects can be generalized/specialized to a third domain. Both can be expressed using the mechanism of mappings between the domains of the concerned objects as follows :
  - Generalization can be expressed as a total, many-one mapping from the union of the domains of the objects being generalized to the domain of the generalized object.
  - Specialization can be expressed as a total, many-one mapping from the domain of the specialized object to the domain of the object being specialized.

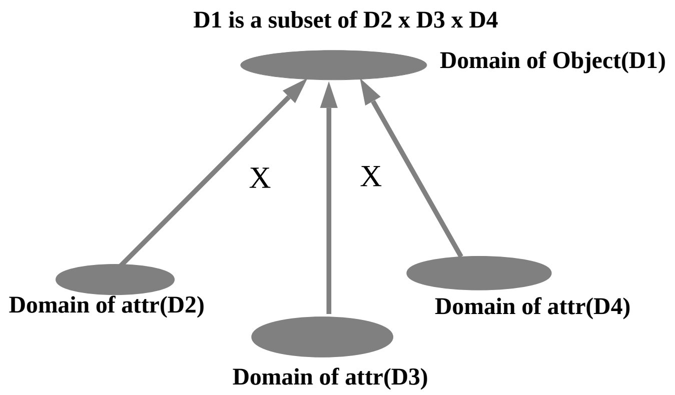

Figure 2: Domain of an Object and it's Attributes

- The **aggregation** abstraction to relate the domains of the objects. This can be expressed as a partial, 1-1 mapping between the cross-product of the domains of the objects being aggregated and the domain of the aggregated object.
- **Functional Dependencies**. They can be expressed as a partial, many-one mapping between the cross-products of the domains of the determining objects and the cross-product of the domains of the determined objects.
- **ANY**. This is used to denote that any abstraction such as the ones defined above may be used to define a mapping between two objects.
- **NONE**. This is used to denote that there is no mapping defined between two semantically related objects.
- **NEG**. This is used to denote that there is no mapping possible between two semantically unrelated objects.

2.1.3 Domains of the Objects

Domains refer to the sets of values from which the objects can take their values. When using an object-oriented model, the domains of objects can be thought of as types, whereas the collections of objects might themselves be thought of as classes. A domain can be either **atomic** (i.e., cannot be decomposed any further) or composed of other atomic or composite domains. The domain of an object can be thought of as a subset of the cross-product of the domains of the properties of the object. Analogously, we can have other combinations of domains, viz. union and intersection of domains.

An important distinction between a context and a domain should be noted. One of the ways to specify a context is as a named collection of the domains of objects, i.e. it is associated with a group of objects. A domain on the other hand is a property of an object and is associated with the description of that object.

2.1.4 States (extensions) of the Objects

The state of an object can be thought of as an extension of an object recorded in a database or databases. However, this extension must not be confused with the actual state of the entity (according to the Real World Semantics) being modeled. Two objects having different extensions can have the same state Real World Semantics (and hence be semantically equivalent).

We now use the above model to define a semantic taxonomy consisting of various types of semantic similarities between the objects.

2.2 Semantic Equivalence

This the strongest measure of semantic proximity two objects can have. Two objects are defined to be *semantically equivalent* when they represent the same real world entity or concept. Expressed in our model, it means that given two objects O_1 and O_2, it should be possible to define a total 1-1 value mapping between the domains of these two objects in any context. Thus we can write it as:

semPro(O_1, O_2) = <ALL, total 1-1 value mapping, (D_1, D_2), _>[3]

The notion of equivalence described above depends on the definition of the domains of the objects and can be more specifically called *domain semantic equivalence*. We can also define a stronger notion of semantic equivalence between two objects which incorporates the state of the databases to which the two objects belong. This equivalence is called *state semantic equivalence*, and is defined as:

semPro(O_1, O_2) = <ALL, M, (D_1, D_2), (S_1, S_2) >
where M is a total 1-1 value mapping between (D_1, S_1) and (D_2, S_2).

Unless explicitly mentioned, we shall use semantic equivalence to mean domain semantic equivalence.

2.3 Semantic Relationship

This is a weaker type of semantic similarity than semantic equivalence. Two objects are said to be *semantically related* when there exists a partial many-one value mapping, or a generalization, or aggregation abstraction between the domains of the two objects. Here we relax the requirement of a 1-1 mapping in a way that given O_1 we can identify O_2 but not vice versa. The requirement that the mapping be definable in any context is not relaxed. Thus we can define the *semantic relationship* as:

semPro(O_1, O_2) = <ALL, M, (D_1, D_2), _>
where M = partial many-one value mapping, generalization, or aggregation

[3]We use the "_" sign to denote don't care.

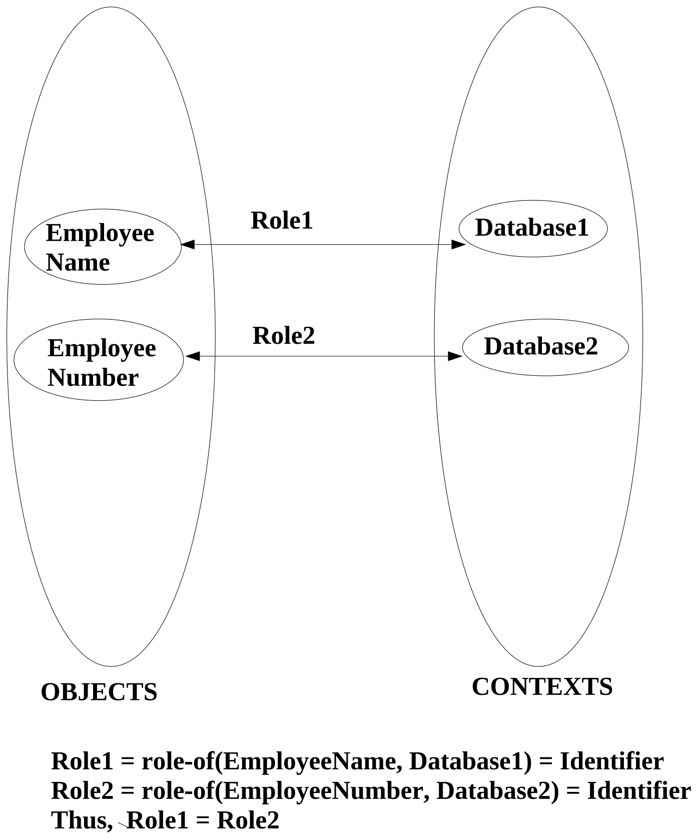
Employee Name
Role1
Database1
Employee Number
Role2
Database2
OBJECTS
CONTEXTS
Role1 = role-of(EmployeeName, Database1) = Identifier
Role2 = role-of(EmployeeNumber, Database2) = Identifier
Thus, Role1 = Role2

Figure 3: Roles played by objects in their contexts

2.4 Semantic Relevance

We consider two objects to be *semantically relevant* if they can be related to each other using some *abstraction* in the *same context*. Thus the notion of semantic relevance between two objects is context dependent, i.e., two objects may be semantically relevant in one context, but not so in another. Objects can be related to each other using any abstraction.

semPro(O_1, O_2) = <SAME, ANY, (D_1, D_2), _>

2.5 Semantic Resemblance

This is the weakest measure of semantic proximity, which might be useful in certain cases. Here, we consider the case where the domains of two objects cannot be related to each other by any abstraction in any context. Hence, the exact nature of semantic proximity between two objects is very difficult to specify. In this case, the user may be presented with extensions of both the objects. In order to express this type of semantic similarity, we introduce an aspect of context, which we call **role**, by extending the concept of role defined in [EN89]. Semantic resemblance is defined in detail in section 2.5.2

2.5.1 Role played by an Object in a Context

This refers to the relationship between an object and the semantic context to which it belongs. We characterize this relationship as a binary function, which has the object and it's context as the arguments and the name of the role as the value.

$$\text{role : object} \times \text{context} \rightarrow \text{rolename}$$

The mapping defined above may be multi-valued, as it is possible for an object to have multiple roles in the same context. However, for our purposes, we shall assume the mapping to be a single-valued binary function.

2.5.2 Roles and Semantic Resemblance

Whenever two objects cannot be related to each other by any abstraction in any context, but they have the same roles in their respective context(s) (where the respective contexts may or may not be the same), they can be said to *semantically resemble* each other. This is a generalization of DOMAIN-DISJOINT-ROLE-EQUAL concept in [LNE89].

semPro(O_1, O_2) = <context, NONE, (D_1, D_2), _>
where context = context(O_1) $\cup$ context(O_2)
and $D_1 \neq D_2$
and role-of(O_1, context) = role-of(O_2, context)

Example : In this example we demonstrate the semantic aspect of the similarity between two objects captured by context. It is possible for two objects to be semantically closer in one context as compared to another context. Thus, it is possible for the same structural schema to have different semantic similarities.

```
Consider two objects:

OBJ1 = TELECOM-EMPLOYEE(ID, SALARY, ....)
OBJ2 = BANK-EMPLOYEE(ID, SALARY, ....)

Suppose the IRS (a government income tax department) wants to query both
these objects wrt the tax bracket they fall in. OBJ1 and OBJ2 can be defined
to be semantically relevant using the following information:

context = context(OBJ1) = context(OBJ2) = IRS (i.e., context = SAME)
abstraction : EMPLOYEE(ID, SALARY) = generalize(OBJ1, OBJ2)

What if there is no single context, such as the one needed for the IRS
application, in which the above objects are to be considered ? Should
the objects then, be considered for schema integration ?
The weaker semantic proximity of semantic resemblance can be defined
between OBJ1 and OBJ2 using the following information:
```

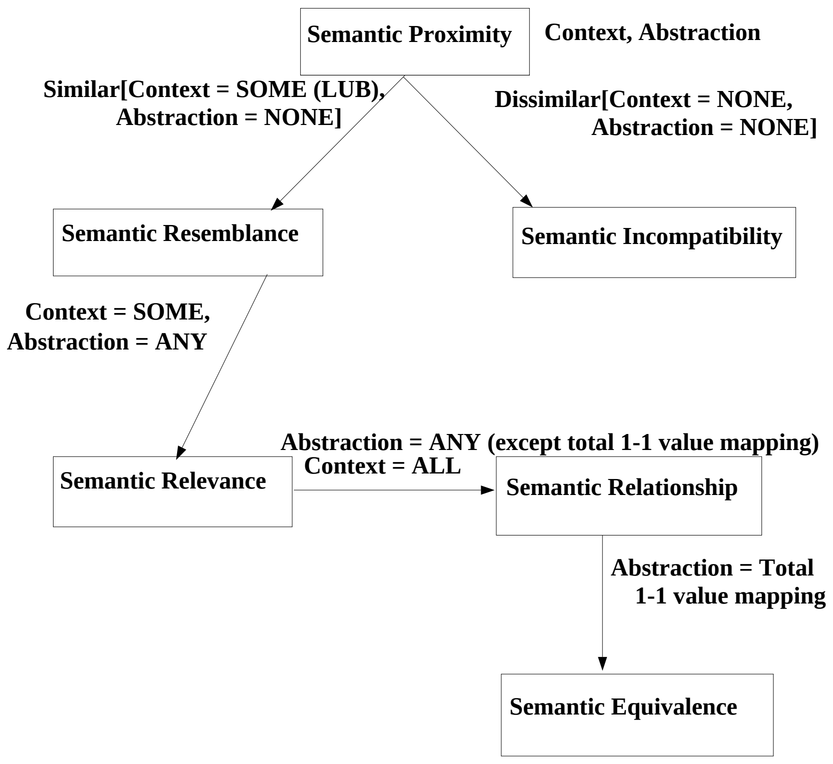
Semantic Proximity
Context, Abstraction
Similar[Context = SOME (LUB), Abstraction = NONE]
Dissimilar[Context = NONE, Abstraction = NONE]
Semantic Resemblance
Semantic Incompatibility
Context = SOME, Abstraction = ANY
Semantic Relevance
Abstraction = ANY (except total 1-1 value mapping) Context = ALL
Semantic Relationship
Abstraction = Total 1-1 value mapping
Semantic Equivalence

Figure 4: Semantic Classification of Object Similarities

```
context(OBJ1) = TELECOMMUNICATION
context(OBJ2) = BANKING
role-of(OBJ1, TELECOMMUNICATION) = SUBORDINATE
role-of(OBJ2, BANKING) = SUBORDINATE
```

2.6 Semantic Incompatibility

While all the proximity measures defined above describe semantic similarity, semantic incompatibility asserts semantic dissimilarity. Lack of any semantic similarity does not automatically imply that the objects are semantically incompatible. Establishing semantic incompatibility requires asserting that there is no context and no abstraction in which the domains of the two objects can be related. Furthermore, the two objects cannot have similar roles in the context(s) in which they exist.

semPro(O_1, O_2) = <NONE, NEG, (D_1, D_2), _>
where context = context(O_1) $\cup$ context(O_2) is undefined,
and Abstraction = NEG, signifying the dissimilarity
and D_1 may or may not be equal to D_2
and role-of(O_1, context) and role-of(O_2, context) are incomparable

3 Semantic Proximity and Uncertainty Modeling

Specifying object relationships involve determining equivalence or subtype assertions between schema objects (e.g., entities) and between attributes. One approach has been to group attributes in a taxonomy of equivalence classes [ELN86, ME84] or a subtype hierarchy [SG89][SM92] and specify object relationships based on assertions among attributes. Another approach has been to annotate attributes or objects with a set of concepts from a global concept space [YSDK91] and determine the object relationships based on the concepts they are related to. Whether giving assertions among attributes or concepts, in practice we often can give only a fuzzy (i.e., uncertain or ambiguous) assertion. Being able to model uncertainty can help in identifying a larger class of assertions leading to better identification of semantic proximities among the objects.

3.1 Previous approaches to model uncertainty

There have been attempts to model uncertainty in the relationships between objects. One approach has been to determine the similarity of objects by utilizing fuzzy and incomplete terminological knowledge [FKN91] together with schema knowledge. The difficulty of this approach is that the assignment of fuzzy strengths is based on intuition, and albeit arbitrary. We are of the opinion that such an assignment of certainty measures is a context sensitive process and depends on the relation between the domains of the terminological entities involved. Thus, these factors could form a basis for assignment of the fuzzy strengths.

Another approach has been that of using partial values and maybe tuples [DeM89]. In this approach, the partial and maybe information has a more formal basis, i.e., a value mapping between the domains of the objects. In our opinion fuzzy logic gives us a more complete framework than the 3-valued logic used to denote the maybe tuples, to represent the full range of uncertain information. Mapping information as a basis for determining uncertainty is inadequate in many cases and in such cases, using the context and the extensions of the objects can be helpful.

Example :

```
Consider two objects STUDENT and DEPARTMENT defined as follows
                 STUDENT(Id#, Name, Grade)
                 DEPARTMENT(Num, Name, Address)
Let Domain(STUDENT.Id#) = {123, 456, 789}
and Domain(DEPARTMENT.Num) = {321, 654, 987}

It is possible to define a Mapping between the domains defined above,
but this does not mean that STUDENT.Id# is equivalent to DEPARTMENT.Num
```

A third approach [BGMP90, TCY93] has used discrete probability distributions to model uncertainty. However, probability values are either assigned as a measure of belief or by an analysis of the underlying sample. If the values are assigned as a measure of belief as in [Zad78], then there should be specified an underlying basis for specifying these measures. Also, if these values are assigned by analyzing the underlying sample, then they depend on the extensions or state of the objects, which might be rendered obsolete in a continually changing database. The *implicit independence assumption* which says that the probabilities modeling the uncertainty of an attribute are independent of the probability values of the other attributes does not appear to accurately reflect the Real World Semantics.

Example :

```
Consider two objects INSTR1 and INSTR2 as defined below
              INSTR1(SS#, HPhone, OPhone)
              INSTR2(SS#, Phone)
in two different databases.
```

M_1 : INSTR1.HPhone $\rightarrow$ INSTR2.Phone
M_2 : INSTR1.OPhone $\rightarrow$ INSTR2.Phone
be two mappings define between the attributes of the objects.

Obviously M_1 and M_2 are not independent of each other and are related to each other through the mappings
M_3 : INSTR1.SS# $\times$ INSTR1.HPhone $\rightarrow$ INSTR2.SS# $\times$ INSTR2.Phone
M_4 : INSTR1.SS# $\times$ INSTR1.OPhone $\rightarrow$ INSTR2.SS# $\times$ INSTR2.Phone

We propose representing the uncertainty in the integration assertions by using the concept of *semantic proximity* defined in the previous section. We also show how the semantic proximities can provide a well defined basis for the assignment of fuzzy strengths. We also show how heuristics used to assign the fuzzy strengths can be simulated using the semantic proximity as the basis.

3.2 Fuzzy Strengths as a function of Semantic Proximity

In this section we establish the semantic proximity as a basis for the assignment of fuzzy strengths to the terminological relationships between two semantically similar objects. As noted in the previous section, when we assign fuzzy strengths to semantic similarities between schema objects, they should reflect the Real World Semantics. Thus any such assignment of belief measures should depend on and reflect :

- The **context(s)** to which the two schema objects belong to.
- The **mapping(s)** which may exist between the domains of the objects or the domains of the individual attributes of the objects. Here, it may be noted that the mappings between two attributes of the objects might not be independent of each other, but maybe dependent. Thus, instead of having mappings $A_{1,1} \rightarrow A_{2,1}$ and $A_{1,2} \rightarrow A_{2,2}$, where $A_{i,j}$ is the jth attribute of the ith object, we might have mappings between pairs of attributes, i.e. $A_{1,1}\times A_{1,2} \rightarrow A_{2,1}\times A_{2,2}$. Hence, the implicit independence assumption of [BGMP90] might not accurately reflect the mappings.

- The **state(s)** or the extensions of the two objects.

The semantic proximity described in the previous section is able to capture this information which represents the semantic similarity between two objects according to the Real World Semantics. Also the interactions between any two attributes of an object can be captured using the interactions between the mappings of the two attributes, thus avoiding the need for the implicit independence assumption.

We define an **uncertainty function** μ between two objects O_1 and O_2 which maps the semantic proximity to the real interval [0,1]. Thus

μ : semPro(O_1, O_2) $\rightarrow$ [0,1]
i.e., μ(Context, Abstraction, (D_1, D_2), (S_1, S_2)) = X where $0 \leq X \leq 1$.

μ is a **user defined** function such that it accurately reflects the Real World Semantics and may not have specific mathematical properties. It may or may not be a computable function. If it is a computable function, that would mean that we can automate the process of assigning the fuzzy strengths to the semantic relations between schema objects. However, it would require the semantic proximities discussed earlier. Two users might choose to define the function differently, but now we have a basis on which to judge, given the semantic proximity, which function is a better reflection of the Real World Semantics. If μ is not computable, a human makes an assignment based on the context(s), the mapping(s) between the domains of the two objects, and possibly the states of the two objects.

Earlier we defined the various kinds of semantic proximities. Now, based on these semantic proximities, we develop a *bounded correctness criterion* which any user defined uncertainty function should follow.

Bounded correctness criterion

Given a user defined uncertainty function μ, let the values to which it maps the various semantic proximities be given as follows :

μ(State-Equivalent) = $X_{StateEq}$
μ(Domain-Equivalent) = X_{DomEq}
μ(Related) = X_{Relat}
μ(Relevant) = X_{Relev}
μ(Resemble) = X_{Res}
μ(Incompatible) = X_{In}

A criterion that a heuristic may meet to justify consistent derivation of the fuzzy strengths is the bounded correctness criteria specified as follows :

1. $X_{StateEq} = 1$
2. $0 < X_{Res} \leq X_{Relev} \leq X_{Relat} \leq X_{DomEq} < 1$
3. $X_{In} = 0$

3.3 Simulation of heuristics using semantic proximity for assignment of Fuzzy Strengths

In this section we debate whether the definition of the user defined uncertainty function μ described in the previous section can capture all the heuristics used to assign fuzzy measures. We show how some of the proposed heuristics used for assignment of fuzzy strengths to the relationships can be simulated using the semantic proximities and by defining an appropriate uncertainty function.

3.3.1 The heuristic of % of common attributes

This is a very simple heuristic [ELN86] to represent using the semantic proximity. It is a heuristic which essentially exploits the *structural similarity* between two entities. The uncertainty function will be independent of the context(s) and the states of the objects. Given the semantic proximity, the uncertainty function μ can be defined as follows :

$\mu(\text{Contexts}, \{\text{M}_i\}, (\{\text{D}_{1,i}\}, \{\text{D}_{2,i}\}), (\text{S}_1, \text{S}_2)) = \frac{|\{D_{1,i}\}|\times 100}{|\{attr(O_1)\}|}$

where M_i is a mapping between the domains of the ith attribute of the two objects and can be represented as
$\text{M}_i : \text{D}_{1,i} \rightarrow \text{D}_{2,i}$.

Example : Consider two Union Incompatible entities as follows :

$\text{Student}_1(\text{Id\#, Name, Grade})$
$\text{Student}_2(\text{Id\#, Name, Address})$

The semantic proximity can be given as :
$\text{semPro}(\text{Student}_1, \text{Student}_2)$
$= <\text{ALL}, \{\text{M}_{Id\#}, \text{M}_{Name}\}, (\{\text{D}_{1,Id\#}, \text{D}_{1,Name}\}, \{\text{D}_{2,Id\#}, \text{D}_{2,Name}\}), \_>$

The uncertainty function is then given as :
$\mu(\text{Student}_1, \text{Student}_2) = \frac{|\{D_{1,Id\#}, D_{1,Name}\}|\times 100}{|\{Id\#, Name, Grade\}|}$

3.3.2 The heuristic of instance participation

This heuristic uses the concept of the cardinality constraints of the entities participating in the mappings [EN89, VH91] to define the uncertainty function. Also, though this function expresses more semantic information than the previous one, it is independent of the context(s) of the two objects. Thus we can define the uncertainty function, for a semantic proximity with the cardinality constraints of the objects participating in the mappings, as follows :

Let O_1 and O_2 be two schema objects and let their semantic proximity be given as follows :

$\text{semPro}(\text{O}_1, \text{O}_2) = <\text{ALL}, \text{Abstraction}, (\text{D}_1, \text{D}_2), \_>$

where Abstraction is a total many-one value mapping between the domains with the cardinality constraints of the domains participating in the mapping given as :
$D_1 \rightarrow (\text{min}_1, \text{max}_1)$ and $D_2 \rightarrow (\text{min}_2, \text{max}_2)$
where min_i and max_i are the minimum and maximum number of elements of domain D_i participating in the mappings.
$\mu(\text{Contexts, Abstraction}, (D_1, D_2), (S_1, S_2)) = \frac{(min_1+max_1)(min_2+max_2)}{4 \times max_1 \times max_2}$
@

4 Domain Incompatibility Problem

In this section we discuss the incompatibilities that arise between two objects when they have differing definitions of semantically similar attribute domains. A broad definition of this incompatibility was given in [CRE87]. We examine in detail the aspects in which two attribute domain definitions can differ and give a comprehensive enumeration of the resulting types of incompatibilities. For each enumerated conflict, we identify the likely semantic proximities between the domains.

4.1 Naming Conflicts

Two attributes that are semantically alike might have different names. They are known as *synonyms*.

Example :
Consider two databases having the relations :

```
STUDENT(Id#, Name, Address)
TEACHER(SS#, Name, Address)
```

`STUDENT.Id# and TEACHER.SS# are synonyms.`

Mappings between synonyms can often be established *wrt* all contexts. In such cases, two objects O_1 and O_2 can be considered to be *semantically equivalent*.

Two attributes that are semantically unrelated might have the same names. They are known as *homonyms*.

Example :
Consider two databases having the relations :

```
STUDENT(Id#, Name, Address)
BOOK(Id#, Name, Author)
```

`STUDENT.Id# and BOOK.Id# are homonyms.`

Since homonyms are semantically unrelated, there cannot be any context, in which there is an abstraction which maps one homonym to another. In such cases, two objects O_1 and O_2 can be considered to be *semantically incompatible*.

4.2 Data Representation Conflicts

Two attributes that are semantically similar might have different data types or representations.

Example :

STUDENT.Id# is defined as a 9 digit integer.
TEACHER.SS# is defined as an 11 character string.

Conversion mappings or routines between different data representations can often be established *wrt* all contexts. In such cases, two objects O_1 and O_2 can be considered to be *semantically equivalent.*

4.3 Data Scaling Conflicts

Two attributes that are semantically similar might be represented using different units and measures. There is a one-one mapping between the values of the domains of the two attributes. For instance, the salary attribute might have values in \$ and £ .

Typically mappings between data represented in different scales can be easily expressed in terms of a function or a lookup table, or by using dynamic attributes as in [LA86] and *wrt* all contexts. In such cases, two objects O_1 and O_2 can be considered to be *semantically equivalent.*

4.4 Data Precision Conflicts

Two attributes that are semantically similar might be represented using different precisions. This case is different from the previous case in that there may not be one-one mapping between the values of the domains. There may be a many-one mapping from the domain of the precise attribute to the domain of the coarse attribute.

Example

Let the attribute Marks have an integer value from 1 to 100.
Let the attribute Grades have the values {A, B, C, D, F}.

There may be a many-one mapping from Marks to Grades. Grades is the coarser attribute.

Typically, mappings can be specified from the precise data scale to the coarse data scale *wrt* all contexts. The other way round, e.g., given a letter grade identifying the precise numerical score, is typically not possible. In such cases, two objects O_1 and O_2 can be considered to have a *semantic relationship.*

4.5 Default Value Conflicts

This type of conflict depends on the definition of the domain of the concerned attributes. The *default value* of an attribute is that value which it is defined to have in the absence

Marks	Grades
81-100	A
61-80	B
41-60	C
21-40	D
1-20	F

Table 1: Mapping between Marks and Grades

of more information about the real world. These conflicts were discussed in [KS91] and can be classified as the broader class of domain incompatibility conflicts. In this case, two attributes might have different default values in different databases. For instance, the default value for Age of an adult might be defined as 18 years in one database and as 21 years in another.

It may not be possible to specify mappings between a default value of one attribute to the default value of another in all contexts. However, it is often possible to define a mapping between them *wrt* the same context. In such cases, the two objects O_1 and O_2 can be considered to be *semantically relevant*, i.e., their *semantic proximity* can be defined as follows :

semPro(Age_1, Age_2) = <SAME, Abstraction, (D_1, D_2), _>
Context = SAME = LegalDriver for Age_1 and Age_2
Abstraction = 1-1 value mapping

4.6 Attribute Integrity Constraint Conflicts

Two semantically similar attributes might be restricted by constraints which might not be consistent with each other. For instance, in different databases, the attribute Age might follow these constraints :

Example :

C1 : Age $\leq$ 18
C2 : Age $>$ 21
C1 and C2 are inconsistent and hence the integrity constraints on the attribute Salary are said to conflict.

Depending on the nature of the integrity constraints involved, it might be possible to generalize the constraints and have a mapping from the specific to the general constraints. However, in certain cases the nature of inconsistency might be such that a mapping might not be possible. Even in that case, the objects O_1 and O_2 can be considered to *semantically resemble* each other, if they have the same role in their respective context(s).

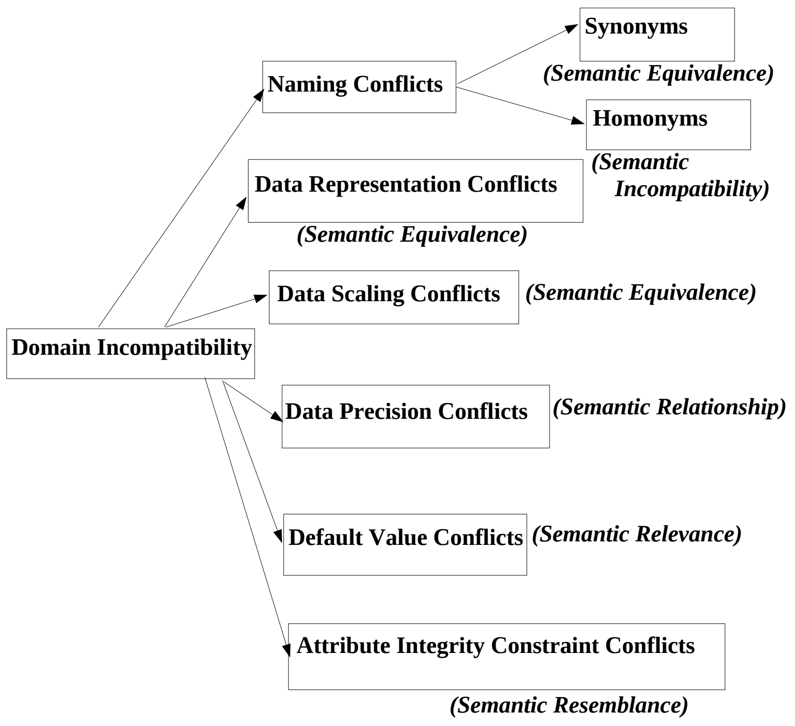

Figure 5: Domain Incompatibility and the likely types of semantic proximities

semPro(Age_1, Age_2) = <context, NONE, (D_1, D_2), _>
where context = context(Age_1) $\cup$ context(Age_2)
and $D_1 \neq D_2$
and role-of(Age_1, context) = role-of(Age_2, context) = AGE

5 Entity Definition Incompatibility Problem

In this section we discuss the incompatibilities that arise between two objects when the entity descriptors used by the objects are only partially compatible, even when the same type of entity is being modeled. The broad definition of this class of conflicts was given in [CRE87]. Here we examine in detail the scenarios in which the entity definitions of semantically similar entities might conflict to give a more precise and comprehensive enumeration of the above class of conflicts. For each enumerated conflict, we identify the likely semantic proximities between the entities.

5.1 Database Identifier Conflicts

In this case, the entity descriptions in two databases are incompatible because they use identifier records that are semantically different. In a relational model scenario, this would translate to two relations modeling the same entity having semantically different keys. This is also known as the *key equivalence problem.*

Example :

```
                  STUDENT1(SS#, Course, Grades)
                  STUDENT2(Name, Course, Grades)
STUDENT1.SS# and STUDENT2.Name are semantically different keys.
```

The semantic proximity of objects having this kind of conflict depends on whether it is possible to define an abstraction to map the keys in one database to another. However, if we assume that the context(s) of the identifiers are defined in the local schemas, we know that they play the *role* of *identification* in their respective contexts. Hence, the weakest possible measure of *semantic proximity* applies, though stronger measures might apply too. The *semantic resemblance* between the above two objects can be defined as :

semPro(O_1, O_2) = <$LS_1 \cup LS_2$, _ , (D_1, D_2), _>
where D_1 = Domain(key(O_1))
and D_2 = Domain(key(O_2))
and role-of(key(O_1), LS_1) = role-of(key(O_2), LS_2) = IDENTIFIER

5.2 Naming Conflicts

Semantically alike entities might be named differently in different databases. For instance, EMPLOYEE and WORKERS might be two objects describing the same set of entities. They are known as *synonyms* of each other. Typically, mappings between synonyms can often be established. In such cases objects O_1 and O_2 having this kind of a conflict can be considered to be *semantically equivalent.*

On the other hand, semantically unrelated entities might have the same name in different databases. For instance, TICKETS might be the name of a relation which models movie tickets in one database, whereas it might model traffic violation tickets in another database. They are known as *homonyms* of each other. Since homonyms are semantically dissimilar, there cannot be any context, in which there is an abstraction which maps one homonym to another. Thus two objects O_1 and O_2 having this conflict can be considered to be *semantically incompatible.*

Note that the above conflicts are different from the *Naming Conflicts* discussed in Section 4.1 of this paper. The conflicts discussed in Section 4.1 arise due to differences in the naming of *attributes* whereas, conflicts in this section arise due to differences in the naming of *entities.*

5.3 Union Compatibility Conflicts

Descriptors of semantically similar entities might not be union compatible with each other. Two entities are union incompatible when the set of attributes are semantically unrelated in such a way that a one-one mapping is not possible between the two sets of attributes.

Example :

```
STUDENT1(Id#, Name, Grade)
```

```
                    STUDENT2(Id#, Name, Address)
are two entities that are union incompatible.
```

Since mappings can be established between the objects on the basis of the common and identifying attributes, objects O_1 and O_2 can be considered to have a *semantically relationship*, i.e. their *semantic proximity* can be defined as follows:

semPro(O_1, O_2) = <ALL, {M_{ID}, M_i}, ({$D_{1,ID}$, $D_{1,i}$}, {$D_{2,ID}$, $D_{2,i}$}), _>
where M_{ID} is a total 1-1 value mapping between the identifiers of the two objects $D_{1,ID}$ and $D_{2,ID}$,
M_i may be a total/partial 1-1/many-one value mapping between $D_{1,i}$ and $D_{2,i}$ and represents the mapping between the ith attribute which is common between the two objects.

5.4 Schema Isomorphism Conflicts

Semantically similar entities may have different number of attributes, giving rise to schema isomorphism conflicts.

Example :

```
                INSTRUCTOR1(SS#, HomePhone, OffPhone)
                INSTRUCTOR2(SS#, Phone)
is an example of schema non-isomorphism.
```

It should be noted that this can be considered an artifact of the *Data Precision Conflicts* identified in section 4.4 of this paper, as the Phone number of INSTRUCTOR1 can be considered to be represented in a more precise manner than the Phone number of INSTRUCTOR2. However, the conflicts discussed in section 4.4 are due to differences in the domains of the attributes representing the same information and hence are *attribute level conflicts*. Whereas, conflicts in this sections arise due to differences in the way the entities INSTRUCTOR1 and INSTRUCTOR2 are defined in the two databases and hence are *entity level conflicts*.

Since mappings can be established between the objects on the basis of the common and identifying attributes, objects O_1 and O_2 can be considered to have a *semantic relationship*, i.e. their *semantic proximity* can be defined as follows:

semPro(Instructor$_1$, Instructor$_2$)
= <ALL, {M_{ID}, M_1}, ({$D_{1,ID}$, $D_{1,2}$, $D_{1,3}$}, {$D_{2,ID}$, $D_{2,2}$}), _>
where M_{ID} is a total 1-1 value mapping between $D_{1,ID}$ and $D_{2,ID}$ and represents the mapping between the identifiers of the two objects.
M_1 may be a total/partial 1-1/many-one value mapping between $D_{1,2} \cup D_{1,3}$ and $D_{2,2}$.

5.5 Missing Data Item Conflicts

This conflict arises when of the entity descriptors modeling semantically similar entities, one has a missing attribute. This type of conflict is subsumed by the conflicts discussed before.

There is a special case of the above conflict which satisfies the following conditions :
The missing attribute is compatible with the entity, and
There exists an inference mechanism to deduce the value of the attribute.

Example :

```
                    STUDENT(SS#, Name, Type)
                    GRAD-STUDENT(SS#, Name)
STUDENT.Type can have values "UG" or "Grad"
GRAD-STUDENT does not have a Type attribute, but that can be implicitly
deduced to be "Grad".
```

It should be noted that in the above example, GRAD-STUDENT can be thought to have a Type attribute whose default value is "Grad". The conflict discussed in this section is different from the *default value* conflict in section 4.5 which is an *attribute level conflict.* A potential resolution of the conflict discussed in this section which is an *entity level conflict* is based on the default value aspect of the *attribute level conflict* of section 4.5.

In this case, a mapping is possible between the objects, only after the value of the missing data item has been deduced. Hence, the process of deduction itself may be viewed as a mapping process. It is always possible to deduce a mapping *wrt* a context. Hence any two objects O_1 and O_2 having this kind of a conflict can be considered *semantically relevant.*

In the above example, before we are able to map the domains of the Type attributes in the two databases, we might have to use the generalization abstraction as follows :

Student = Generalize(GRAD-STUDENT)

and then we can introduce a partial 1-1 value mapping between the default values of the missing attribute(s).

semPro(STUDENT, GRAD-STUDENT)
= <SAME, Abstraction, (D_1, D_2), _>
where Abstraction = Generalization $\circ$ partial 1-1 value mapping
and Context = SAME = *wrt* which the mapping has been deduced
and D_1 = {"UG", "Grad"} and D_2 = {"Grad"}

6 Data Value Incompatibility Problem

This class of conflicts covers those incompatibilities that arise due to the values of the data present in different databases [BOT86]. This class of conflict is different from the default value conflicts and attribute integrity constraint conflicts described in Section 4. The latter type of conflict is due to the definitions of the values of the attribute domains, whereas here we refer to the data values already existing in the database. Thus, the conflicts here depend on the database state. Since we are dealing with independent databases,

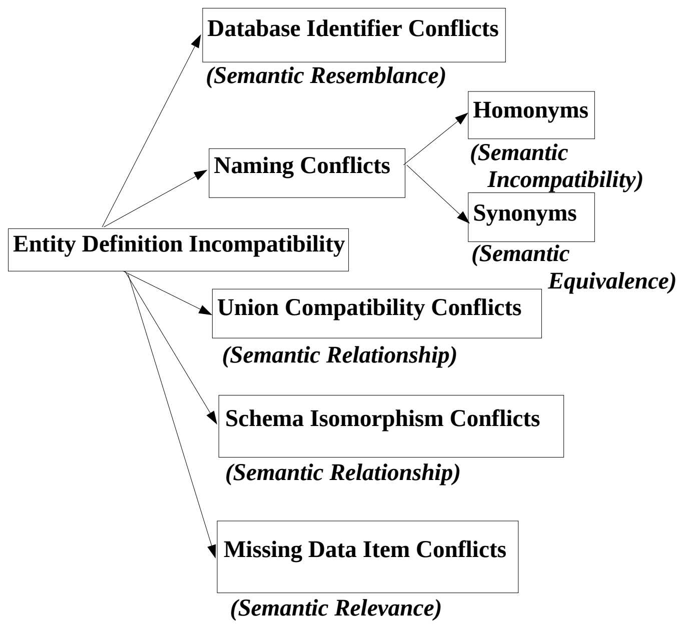
Database Identifier Conflicts
(Semantic Resemblance)
Homonyms
(Semantic Incompatibility)
Naming Conflicts
Synonyms
(Semantic Equivalence)
Entity Definition Incompatibility
Union Compatibility Conflicts
(Semantic Relationship)
Schema Isomorphism Conflicts
(Semantic Relationship)
Missing Data Item Conflicts
(Semantic Relevance)

Figure 6: Entity Definition Incompatibilities and the likely types of semantic proximities

it is not necessary that the data values for the same entities in two different databases be consistent with each other.

Example :

```
Consider two databases modeling the entity Ship
                 SHIP1(Id#, Name, Weight)
                 SHIP2(Id#, Name, Weight)
Consider a entity represented in both databases as follows :
                 SHIP1(123, USSEnterprise, 100)
                 SHIP2(123, USSEnterprise, 200)
Thus, we have the same entity for which SHIP1.Weight is not the same as
SHIP2.Weight, i.e., it has inconsistent values in the database.
```

In this section we give a more detailed classification of the data value inconsistencies which can arise based on whether the cause of inconsistency is known and the extent and duration of the inconsistency. Also in the semantic classification of two objects having this class of conflicts, the state component of the semantic proximity descriptor plays an important role because the conflicts here are in the extensions and not the schemas of the two objects.

6.1 Known Inconsistency

In this type of conflict, the cause of inconsistency is known ahead of time and hence measures can be initiated to resolve the inconsistency in the data values. For instance, it might be known ahead of time that one database is more reliable than the other. Here the cause of the inconsistency can be identified and the more reliable database can be used to resolve the inconsistency (e.g., overrule the less reliable database).

When, the cause of inconsistency between objects is known ahead of time, it was possible to establish a mapping between objects having inconsistent values. However, the mappings might be between the (Domain, State) of the two objects. Hence, they may be considered to be *state semantically equivalent*, i.e., their semantic proximity can be defined as follows :

semPro(O_1, O_2) = <ALL, M, (D_1, D_2), (S_1, S_2)>
where M is a total 1-1 value mapping between (D_1, S_1) and (D_2, S_2).

6.2 Temporary Inconsistency

In this type of conflict, the inconsistency is of a temporary nature. This type of conflict has been identified in [RSK91] and has been expressed as a *temporal consistency predicate*[4]. One of the databases which has conflicting values, might have obsolete information. This means that the information stored in the databases is time dependent. It is also possible that the change in information in one database has not yet propagated to the other databases.

In this case, since the inconsistency is only of a temporary nature, the objects may be said to be *eventually semantically equivalent.* In this case the semantic classification between two objects O_1 and O_2 depends on their states as well as time. Here we model the state of an object as a function of time. Thus the *semantic proximity* can be defined as follows :

semPro(O_1, O_2)=<ALL, total 1-1 value mapping, (D_1, D_2), (S_1, S_2)>
where $S_2(t + \Delta t) = S_1(t)$.

6.3 Acceptable Inconsistency

In this type of conflict, the inconsistencies between values from different databases might be within an acceptable range. Thus, depending on the type of query being answered, the error in the values of two inconsistent databases might be considered tolerable. The *tolerance* of the inconsistency can be of a numerical or non numerical nature.

Example : Numerical Inconsistency

QUERY : Find the Tax Bracket of an Employee.
INCONSISTENCY : If the inconsistency in the value of an Employee Income is up to a fraction of a dollar it may be ignored.

Example : Non numerical Inconsistency

[4]Additional information on weaker criteria for consistency can be found in the literature on transaction models (e.g., see [SRK92]).

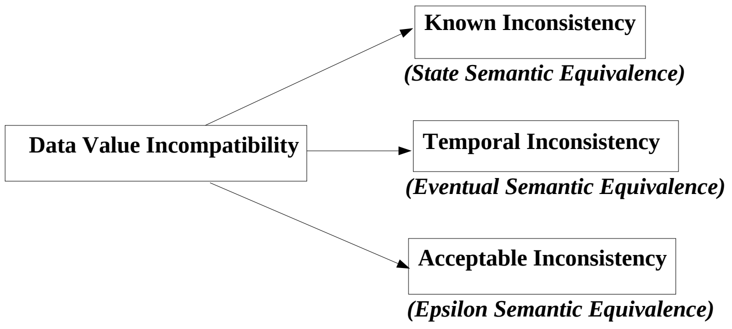
Known Inconsistency
(State Semantic Equivalence)
Data Value Incompatibility
Temporal Inconsistency
(Eventual Semantic Equivalence)
Acceptable Inconsistency
(Epsilon Semantic Equivalence)

Figure 7: Data value incompatibilities and the likely types of semantic proximities

QUERY : Find the State of Residence of an Employee.
INCONSISTENCY : If the Employee is recorded as staying in Edison and New Brunswick (both are in New Jersey), then again the inconsistency may be ignored.

In this case, since the inconsistency between two objects O_1 and O_2 is considered to be acceptable, the two objects may be considered to be *epsilon semantically equivalent*. Thus, the *semantic proximity* can be defined as follows :

semPro(O_1, O_2)=<ALL, total 1-1 value mapping, (D_1, D_2), (S_1, S_2)>
where perturb(S_1, ϵ) = S_2
and ϵ is the discrepancy in the state of the two objects.

7 Abstraction Level Incompatibility Problem

This class of conflicts was first discussed in [DH84] in the context of the functional model. These incompatibilities arise when two semantically similar entities are represented at differing levels of abstraction. Differences in abstraction can arise due to the different levels of generality at which an entity is represented in the database. They can also arise due to aggregation used both at the entity as well as the attribute level.

7.1 Generalization Conflicts

These conflicts arise when two entities are represented at different levels of generalization in two different databases. Also, there might be a natural inclusion relationship induced between the two entities.
Example :

```
Consider the entity "Graduate Students" which may be
represented in two different databases as follows :
                STUDENT(Id#, Name, Major)
                GRAD-STUDENT(Id#, Name, Major, Advisor)
Thus we have the same entity set being defined at a more general
level in the first database.
```

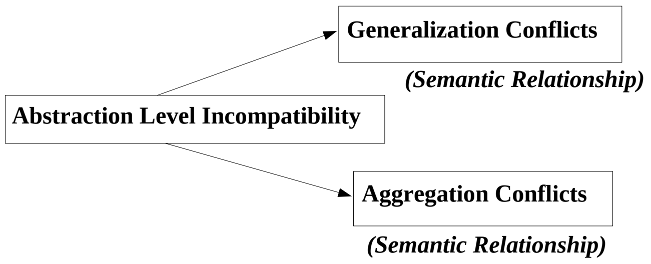

Figure 8: Abstraction level incompatibilities and the likely types of semantic proximities

In this case there is an inclusion relationship between two conflicting objects and hence, they may be considered to have a *semantic relationship*.

semPro(O_1, O_2) = <ALL, Generalization, (D_1, D_2), _>

7.2 Aggregation Conflicts

These conflicts arise when an aggregation is used in one database to identify a set of entities in another database. Also, the properties of the aggregate concept can be an aggregate of the corresponding property of the set of entities.

Example :

```
Consider the aggregation SET-OF which is used to define a concept in the
first database and the set of entities in another database as follows :
                    CONVOY(Id#, AvgWeight, Location)
                    SHIP(Id#, Weight, Location, Captain)
Thus, CONVOY in the first database is a SET-OF SHIPs in the second
database. Also, CONVOY.AvgWeight is the average(aggregate function)
of SHIP.Weight, for every ship that is a member of the convoy.
```

In this case there is a mapping in one direction only, i.e., the an element of a set is mapped to the set itself. In the other direction, the mapping is not precise. When the SHIP entity is known, one can identify the CONVOY entity it belongs to, but not vice versa. Hence two objects might be considered to have a *semantic relationship*. Thus, the *semantic proximity* can be defined as follows :

semPro(O_1, O_2) = <ALL, Aggregation, (D_1, D_2), _>

8 Schematic Discrepancies Problem

This class of conflicts was discussed in [DAODT85, KLK91]. It was noted that these conflicts can take place within the same data model and arise when data in one database correspond to metadata of another database. This class of conflicts is similar to that discussed in Section 6 when the conflicts depend on the database state. We now analyze the problem and identify three aspects with help of an example given in [KLK91].

Example : Consider three stock databases. All contain the closing price for each day of each stock in the stock market. The schemata for the three databases are as follows:

- **Database DB1 :**
  relation r : {(date, stkCode, clsPrice) ... }
- **Database DB2 :**
  relation r : {(date, stk1, stk2, ...) ... }
- **Database DB3 :**
  relation stk1 : {(date, clsPrice) ... },
  relation stk2 : {(date, clsPrice) ... },
  ⋮

DB1 consists of a single relation that has a tuple per day per stock with its closing price. DB2 also has a single relation, but with one attribute per stock, and one tuple per day, where the value of the attribute is the closing price of the stock. DB3 has, in contrast, one relation per stock that has a tuple per day with its closing price. Let us consider that the stkCode values in DB1 are the names of the attributes, and in the other databases they are the names of relations (e.g., stk1, stk2).

8.1 Data Value Attribute Conflict

This conflict arises when the value of an attribute in one database corresponds to an attribute in another database. Thus this kind of conflict depends on the *database state*. Referring to the above example, the values of the attribute *stkCode* in the database *DB1* correspond to the attributes *stk1, stk2*, ... in the database *DB2*.

Since this conflict is dependent on the database state, the fourth component of the 4-tuple describing the semantic proximity plays an important role. Also the mappings here are established between set of attributes ($\{O_i\}$) and values in the extension of the other attribute (O_2). Thus the two objects may be considered to be *meta semantically equivalent* and their *semantic proximity* can be defined as follows :

semPro($\{O_i\}$, O_2) = <ALL, M, (D_1, D_2), (S_1, S_2)>
where M is a total 1-1 mapping between $\{O_i\}$ and S_2.

8.2 Attribute Entity Conflict

This conflict arises when the same entity is being modeled as an attribute in one database and a relation in another database. This kind of conflict is different from the conflicts defined in the previous and next subsections because it depends on the *database schema*

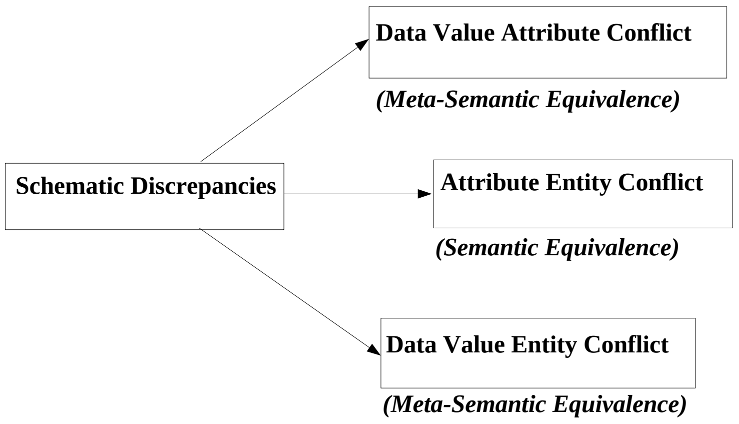
Schematic Discrepancies
Data Value Attribute Conflict
(Meta-Semantic Equivalence)
Attribute Entity Conflict
(Semantic Equivalence)
Data Value Entity Conflict
(Meta-Semantic Equivalence)

Figure 9: Schematic Discrepancies and the likely types of semantic proximities

and not on the *database state*. This conflict can also be classified as a subclass of the **Entity Definition Incompatibility Problem**. Referring to the example described in the beginning of this section the attribute *stk1, stk2* in the database *DB2* correspond to relations of the same name in the database *DB3*.

Objects O_1 and O_2 can be considered to be *semantically equivalent* as 1-1 value mappings can be established between the domains of the attribute (O_1) and the domain of the identifying attribute of the entity (O_2). It should be noted that O_1 is an attribute (property) and O_2 is an entity (object class). Thus the *semantic proximity* can be defined as follows :

semPro(O_1, O_2) = <ALL, total 1-1 value mapping, (D_1, D_2), _>
where D_1 = Domain(O_1)
and D_2 = Domain(Identifier(O_2)).

8.3 Data Value Entity Conflict

This conflict arises when the value of an attribute in one database corresponds to a relation in another database. Thus this kind of conflict depends on the *database state*. Referring to the example described in the beginning of this section, the values of the attribute *stkCode* in the database *DB1* correspond to the relations *stk1, stk2* in the database *DB3*.

Since this conflict is dependent on the database state, the state component of semantic proximity plays an important role. Also the mappings here are established between set of entities ($\{O_i\}$) and values in the extension of an attribute (O_2). Thus the two objects may be considered to be *meta semantically equivalent* and their *semantic proximity* can be defined as follows :

semPro($\{O_i\}$, O_2) = < ALL, M, (D_1, D_2), (S_1, S_2)>
where M is a total 1-1 mapping between $\{O_i\}$ and S_2.

9 Conclusion

An essential prerequisite to achieving interoperability among database systems is to be able to identify relevant data managed by different database systems. This requires us to understand and define the semantic similarities among the objects. We introduced the concept of semantic proximity to specify degrees of semantic similarities among the objects based on their real world semantics, and use it to propose a semantic taxonomy. We also showed how uncertainty measures can be expressed as a function of these semantic proximities. Modeling of several types of inconsistencies is discussed. Thus we establish uncertainty and inconsistency as aspects of semantics.

Building upon earlier work on schematic (structural, representational) differences among objects, we develop a taxonomy of schematic conflicts. A dual semantic *vs* schematic perspective is presented by identifying likely types of semantic similarities between objects with different types of schematic differences.

We are currently developing a uniform formalism to express various schematic conflicts. Additional work is needed to further clarify the nature and structure of the context to which the two objects can belong, as well as the relationship between an object and the context in which the semantic proximity is defined. We also plan to develop a methodology of combining various semantic descriptors. We plan to investigate context dependent uncertainty functions which map semantic proximities to fuzzy strengths.

Acknowledgements

Peter Fankhauser's comments helped us present the uncertainty function μ in the proper perspective.

References

[BGMP90] D. Barbara, H. Garcia-Molina, and D. Porter. A Probabilistic Relational Model. *Lecture Notes in Computer Science : Advances in Database Technology EDBT '90*, #416, 1990.

[BOT86] Y. Breitbart, P. Olson, and G. Thompson. Database Integration in a Distributed Heterogeneous Database System. In *Proceedings of the 2nd IEEE Conference on Data Engineering*, February 1986.

[CRE87] B. Czejdo, M. Rusinkiewicz, and D. Embley. An approach to Schema Integration and Query Formulation in Federated Database Systems. In *Proceedings of the 3rd IEEE Conference on Data Engineering*, February 1987.

[DAODT85] S. Deen, R. Amin, G. Ofori-Dwumfuo, and M. Taylor. The architecture of a Generalised Distributed Database System PRECI*. *IEEE Computer*, 28(4), 1985.

[DeM89] L. DeMichiel. Resolving Database Incompatibility : An approach to performing Relational Operations over Mismatched Domains. *IEEE Transactions on Knowledge and Data Engineering*, 1(4), 1989.

[DH84] U. Dayal and H. Hwang. View definition and Generalization for Database Integration of a Multidatabase System. *IEEE Transactions on Software Engineering*, 10(6), November 1984.

[ELN86] R. Elmasri, J. Larson, and S. Navathe. Schema Integration Algorithms for Federated Databases and Logical Database Design. Technical report, Honeywell Corporate Systems Develpment Division, Golden Valley, MN, 1986.

[EN89] R. Elmasri and S. Navathe. *Fundamentals of Database Systems*. Benjamin/Cummins, 1989.

[FKN91] P. Fankhauser, M. Kracker, and E. Neuhold. Semantic vs. Structural resemblance of Classes. *SIGMOD Record, special issue on Semantic Issues in Multidatabases*, A. Sheth, ed., 20(4), December 1991.

[HM85] D. Heimbigner and D. McLeod. A federated architecture for Information Systems. *ACM Transactions on Office Information Systems*, 3,3, 1985.

[Ken91] W. Kent. The breakdown of the Information Model in Multidatabase Systems. *SIGMOD Record, special issue on Semantic Issues in Multidatabases*, A. Sheth, ed., 20(4), December 1991.

[KLK91] R. Krishnamurthy, W. Litwin, and W. Kent. Language features for Interoperability of Databases with Schematic Discrepancies. In *Proceedings of 1991 ACM SIGMOD*, May 1991.

[KS91] W. Kim and J. Seo. Classifying Schematic and Data Heterogeneity in Multidatabase Systems. *IEEE Computer*, 24(12), December 1991.

[LA86] W. Litwin and A. Abdellatif. Multidatabase Interoperability. *IEEE Computer*, 19(12), December 1986.

[LNE89] J. Larson, S. Navathe, and R. Elmasri. A Theory of Attribute Equivalence in Databases with Application to Schema Integration. *IEEE Transactions on Software Engineering*, 15(4), 1989.

[ME84] M. Mannino and W. Effelsberg. Matching techniques in Global Schema Design. In *Proceedings of the 1st IEEE Conference on Data Engineering*, April 1984.

[PLS92] M. Papazoglou, S. Laufmann, and T. Sellis. An organizational framework for Cooperating Intelligent Information Systems. In *International Journal of Intelligent and Cooperative Information Systems*, March 1992.

[RSK91] M. Rusinkiewicz, A. Sheth, and G. Karabatis. Specifying Interdatabase Dependencies in a Multidatabase Environment. *IEEE Computer*, 24(12), December 1991.

[SG89] A. Sheth and S. Gala. Attribute relationships : An impediment in automating Schema Integration. In *Proceedings of the NSF Workshop on Heterogeneous Databases*, December 1989.

[She91a] A. Sheth. Federated Database Systems for managing Distributed, Heterogeneous, and Autonomous Databases. *Tutorial Notes - the 17th VLDB Conference*, September 1991.

[She91b] A. Sheth. Semantic issues in Multidatabase Systems. *SIGMOD Record, special issue on Semantic Issues in Multidatabases*, A. Sheth, ed., 20(4), December 1991.

[SL90] A. Sheth and J. Larson. Federated Database Systems for managing Distributed, Heterogeneous and Autonomous Databases. *ACM Computing Surveys*, 22(3), September 1990.

[SM91] M. Siegel and S. Madnick. A Metadata Approach to resolving Semantic Conflicts. In *Proceedings of the 17th VLDB Conference*, September 1991.

[SM92] A. Sheth and H. Marcus. Schema Analysis and Integration: Methodology, Techniques and Prototype Toolkit. Technical Report TM-STS-019981/1, Bellcore, March 1992.

[SRK92] A. Sheth, M. Rusinkiewicz, and G. Karabatis. Using Polytransactions to manage Independent Data. In *Database Transaction Models*, 1992.

[TCY93] F. Tseng, A. Chen, and W. Yang. Answering Heterogeneous Database Queries with Degrees of Uncertainty. *Distributed and Parallel Databases, An International Journal*, 1(3), July 1993.

[VH91] V. Ventrone and S. Heiler. Semantic Heterogeneity as a result of Domain Evolution. *SIGMOD Record, special issue on Semantic Issues in Multidatabases*, A. Sheth, ed., 20(4), December 1991.

[Woo85] J. Wood. What's in a link ? In *Readings in Knowledge Representation.* Morgan Kaufmann, 1985.

[YSDK91] C. Yu, W. Sun, S. Dao, and D. Keirsey. Determining relationships among attributes for Interoperability of Multidatabase Systems. In *Proceedings of the 1st International Workshop on Interoperability in Multidatabase Systems*, April 1991.

[Zad78] L. Zadeh. Fuzzy Sets as a basis for a Theory of Possibility. *Fuzzy Sets and Systems*, 1(1), 1978.